

# Шифр гаммирования

---

Андрей Васильев НПИбд-02-19

18 октября, 2022, Москва, Россия

Российский Университет Дружбы Народов

# Цели и задачи

---

# Цель лабораторной работы

Изучение алгоритма шифрования гаммированием

# **Выполнение лабораторной работы**

---

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Наложение (или снятие) гаммы на блок сообщения в рассматриваемом нами стандарте реализуется с помощью операции побитного сложения по модулю 2 (XOR). То есть при шифровании сообщений каждый блок открытого сообщения XORится с блоком криптографической гаммы, длина которого должна соответствовать длине блоков открытого сообщения. При этом, если размер блока исходного текста меньше, чем размер блока гаммы, блок гаммы обрезается до размера блока исходного текста (выполняется процедура усечения гаммы).



Figure 1: Шифрование



Figure 2: Дешифровка



В аддитивных шифрах символы исходного сообщения заменяются числами, которые складываются по модулю с числами гаммы. Ключом шифра является гамма, символы которой последовательно повторяются. Перед шифрованием символы сообщения и гаммы заменяются их номерами в алфавите и само кодирование выполняется по формуле

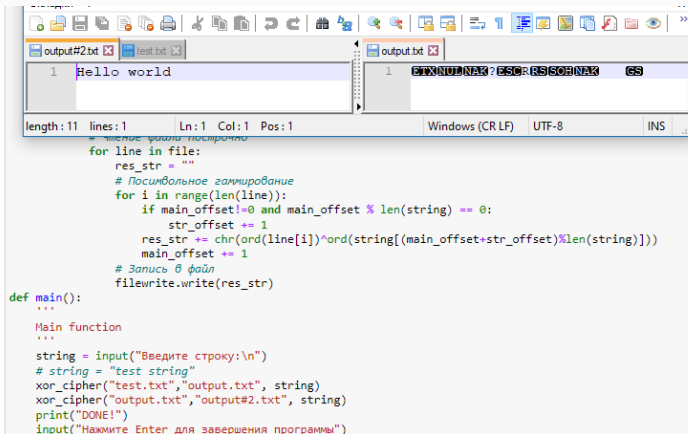
$$C_i = (T_i + G_i) \bmod N$$

# Пример работы алгоритма

<i>T</i>	К	А	Ф	Е	Д	Р	А		С	И	С	Т	Е	М		И	Н	Ф	О	Р	М	А	Т	И	К	И
<i>G</i>	С	И	М	В	О	Л	С	И	М	В	О	Л	С	И	М	В	О	Л	С	И	М	В	О	Л	С	И
<i>T</i>	12	1	22	6	5	18	1	34	19	10	19	20	6	14	34	10	15	22	16	18	14	1	20	10	12	10
<i>G</i>	19	10	14	3	16	13	19	10	14	3	16	13	19	10	14	3	16	13	19	10	14	3	16	13	19	10
<i>T+G</i>	31	11	36	9	21	31	20	44	33	13	35	33	25	24	48	13	31	35	35	28	28	4	36	23	31	20
<i>mod N</i>	31	11	36	9	21	31	20	0	33	13	35	33	25	24	4	13	31	35	35	28	28	4	36	23	31	20
<i>0 → N</i>	31	11	36	9	21	31	20	44	33	13	35	33	25	24	4	13	31	35	35	28	28	4	36	23	31	20
<i>C</i>	Э	Й	1	З	У	Э	Т	9	Я	Л	0	Я	Ч	Ц	Г	Л	Э	0	0	Ъ	Ъ	Г	1	Х	Э	Т

Figure 3: Работа алгоритма гаммирования

# Пример работы программы



The screenshot shows a code editor with two tabs: 'output#2.txt' and 'test.txt'. The 'test.txt' tab is active, displaying the text 'Hello world'. The 'output#2.txt' tab shows the output of a program, which is a series of escape characters: 'ESC X NUL NAK ? ESC R RS SOB NAK CS'. Below the tabs, the status bar indicates 'length: 11 lines: 1', 'Ln: 1 Col: 1 Pos: 1', 'Windows (CR LF)', 'UTF-8', and 'INS'. The main code area contains a Python script with comments in Russian. The script defines a function 'xor\_cipher' and a 'main' function. The 'main' function prompts the user to enter a string, which is then processed by 'xor\_cipher' and the result is written to 'output.txt' and 'output#2.txt'. The script ends with a prompt to press Enter to finish the program.

```
# чтение файла построчно
for line in file:
    res_str = ""
    # Посимвольное гаммирование
    for i in range(len(line)):
        if main_offset != 0 and main_offset % len(string) == 0:
            str_offset += 1
            res_str += chr(ord(line[i]) ^ ord(string[(main_offset + str_offset) % len(string)]))
            main_offset += 1
    # Запись в файл
    filewrite.write(res_str)

def main():
    """
    Main function
    """
    string = input("Введите строку:\n")
    # string = "test string"
    xor_cipher("test.txt", "output.txt", string)
    xor_cipher("output.txt", "output#2.txt", string)
    print("DONE!")
    input("Нажмите Enter для завершения программы")
```

```
In [2]: if __name__ == "__main__":
        main()
```

Введите строку:  
KeyString  
DONE!  
Нажмите Enter для завершения программы

## **Выводы**

---

Изучили алгоритм шифрования с помощью гаммирования